Interactive Data Styling and Multifocal Visualization for a View-Aware Digital Earth

Mark Sherlock, Mahmudul Hasan, and Faramarz Samavati

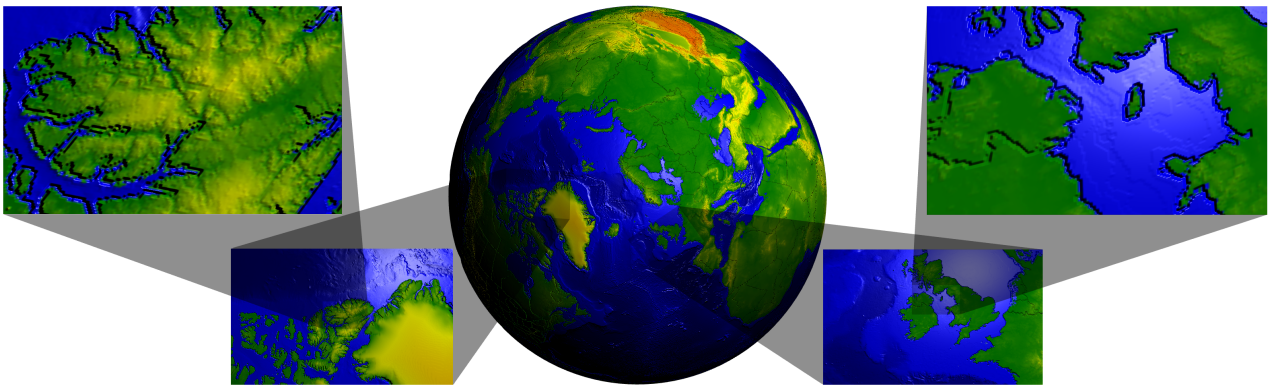


Fig. 1. Interactively created multilevel focus+context hierarchy showing elevation and political boundaries (S1, S2).

Abstract—A Discrete Global Grid System (DGGS) is a powerful tool for creating the discrete reference models that support geospatial dataset integration, organization, processing, and visualization in a Digital Earth (DE) application. However, the growing size and scale of geospatial datasets present significant obstacles to the interactivity and accessibility of geospatial visualizations. To address this challenge, we present a portable DGGS that runs in web browsers on a client device and efficiently communicates with a server-side DGGS. In our method, the client-side is responsible for triggering queries for missing data, managing the viewing area, and rendering various styles and effects. The server is responsible for generating data representations for DGGS cells in response to queries from clients. The resulting system is capable of interactively displaying multiple simultaneous viewpoints which enable support for multilevel focus+context visualization on the globe. We also provide several real-time data styling techniques that are designed to work efficiently on both the client and server. These methods help make DE more accessible and informative than ever before.

Index Terms—earth, data integration, visualization, spatial resolution, context awareness, client-server systems

1 Introduction

A Digital Earth (DE) is a 3D representation of the Earth on which geospatial datasets of any type can be efficiently integrated, organized, and visualized by using the Earth's surface as a reference model [1]. Such representations aim to provide the world's citizens with access, via meaningful interactions and visualizations, to the ever-larger amounts of geospatial data collected daily [2]. It is an important challenge to make such large dynamic multiscale datasets accessible, both in terms of their availability and the in terms of interactive viewing. In this work, we detail approaches to tackle the problem of accessing and displaying multiple large datasets at once. for maximum accessibility, our method targets web browsers, which are now accessible by more than three billion internet users. Thus we introduce a suite of techniques for the interactive web-based visualization of large multiscale datasets for the entire globe.

- *M. Sherlock, M. Hasan, and F. Samavati are with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada. E-mail: {mjsherlo, mhasan, samavati}@ucalgary.ca*

Technical report created October 07, 2016.

One of the main challenges in developing such a system lies in the size of DE datasets, many of which measure in the terabytes. A single operation, such as the rendering of a map, may require the use of several of these datasets at varying resolutions. To address this challenge in a web-based (client-server) environment, we need to solve problems related to memory limitations, computational costs, and network throughput.

Another obstacle to the visualization of DE is the limitation of space, which introduces difficulties in the effective presentation of the large differences in scale, large distances between locations on the globe, and the wide varieties of data available. Viewing two distant cities, for example, at high enough detail level to discern roads is not possible with a single point of view. Furthermore, datasets often have interesting information at multiple levels of detail. Viewing a dataset at city-scale and country-scale simultaneously is often impossible without special visualization techniques. Likewise, there is a limit to the amount of data that can be overlaid on a single view of the globe. We overcome these challenges through the use of multilevel focus+context visu-

alization. Our implementation of focus+context visualization uses multiple live views of the globe in order to concurrently display a variety of locations at separate levels of detail, as shown in Fig. 1. We extend the use of these multiple views to allow for per-focus styling, which allows each individual view of the globe to style, combine, and filter data differently.

In a web-based application, utilizing all required datasets at their native resolution over the entire globe is an extremely slow and resource-intensive approach. Hence, we utilize only what is necessary to accommodate the various view of the globe. For a set of different views, chosen interactively, transmit only those data required by these views, and at resolutions proportional to the size of the viewports, defining the degree of interest (DOI) [3]. Therefore, to capture the importance of this multi-view awareness, we use the analogy of *multi-view aware Digital Earth*. Our DE can rapidly determine which parts of the Earth are needed for visualization (and at what resolution) and queue them for transmission via the web. Views are defined and controlled by virtual cameras, which render their views to a series of viewports. The viewports constitute the nodes in the multilevel focus+context visualization hierarchy.

To create an efficient web-based framework, we divide the required operations between the server and the web browser application (client). An efficient data structure is required to handle the thousands of pieces of data, the geometry of the Earth, and the various textures in play, all of which exist at multiple resolutions. This data structure should also facilitate communication between the server and clients. For this purpose, we use a Discrete Global Grid System (DGGS) [4]. In a DGGS, the Earth is discretized into a hierarchy of highly regular cells with unique indices using a multiresolution tessellation. Commonly, DGGSs define equal area cells due to the ease of statistical analysis on the data assigned to these cells. DGGS provides a powerful reference system for organizing, integrating, and transmitting geospatial data on-the-fly, without the need for GIS experts to process the data [4].

The functionality required of the grid system on the server-side is different from those on the client-side. The DGGS on the server-side is typically more complex and sophisticated, geared towards providing an efficient hierarchical data structure for rapid integration, accurate sampling, and data analytics for data associated with the Earth. On the other hand, the DGGS on the client-side should be streamlined for the tasks of handling user interaction, organizing multiple views, and creating flexible and high-quality renderings for various visualization scenarios. Hence, our method expects the server and clients to use different grid systems, and therefore it is necessary to have an efficient way to convert between these grids.

In our implementation, the server runs off a hexagon-based DGGS [5], [6], which is an appropriate choice for efficient sampling of data. As images come in rectangular formats and rendering pipelines expect square or triangular inputs, it makes sense to convert these hexagonal cells into quadrilaterals before they are transmitted by the server to any clients. This conversion allows for a much more lightweight system on the client, which stores the data in its own quadrilateral-based DGGS. The conversion between these two hierarchical grid systems is efficient and can be performed largely on-the-fly, converting cells as they are requested by the client [7], [8]. Fig. 2 illustrates how these systems interact in our implementation. Such a setup also allows for a high level of interoperability with different flavours of DGGS, as no implementation-specific spherical projections are required on the client.

In order to visualize multiple datasets at once, our choice of a quadrilateral DGGS allows us to encode data as 2D textures – where each pixel in the texture represents a single cell within the DGGS – that can be utilized together during the rendering process in various ways. These textures, which we hereafter refer to as *data textures*, are transmitted as standard images files. We present a method of encoding these data textures, enabling sophisticated rendering techniques and real-time style modification on the client.

Rather than representing colours, each texel of a data texture encodes cell data into the four available channels (RGBA). This is done by sampling the cells within the chosen datasets, normalizing the values, and quantizing them into one or more channels. Up to four datasets can therefore be encoded into a single texture. These textures, when passed to the GPU on the client-side, can be styled in real-time by, e.g., informing the lighting model. An advantage of this system is that it allows the client to alter or animate styling rapidly without the need to request differently-styled data from the server. We further leverage this styling functionality to demonstrate the idea of per-view globe styling, which allows multiple concurrent views of the same globe to produce multiscale visualizations.

In summary, our contributions span several areas of work. We present a novel application of multilevel focus+context visualization to Digital Earth. We show how a DGGS can be extended to work natively within a web-based framework. Finally, our work gives credence to the use of DGGS as a data conversion tool, which allows for the generation of data that is well-suited for rendering regardless of the original data format.

In the domain of visualization, we propose a method to provide client-side data styling with benefits to storage, transmission, and interactivity. This extra styling functionality is provided without significant performance impacts on the server or on the rendering process. We also show the usefulness of fast client-side styling as it enables dataset animation and per-view styling. These visualizations also respect the underlying DGGS framework as the data texture texels correspond to discrete DGGS cells.

This article is arranged as follows. Section 2 covers previous work and related material on DGGS. Section 3 details our approach and implementation. Section 4 discusses the performance characteristics of our method. In Section 5, we display our results. Section 6 concludes this article and acknowledges those who contributed to this work.

2 Related Work

In this section, we review background literature related to the two main aspects of our work: Digital Earth and multilevel focus+context visualization.

2.1 Digital Earth

Digital Earth encapsulates various methods allowing for the storage, integration, processing, visualization, and retrieval

of various types of geospatial data on a virtual globe [1]. One of the most effective tools for supporting a Digital Earth is the Discrete Global Grid System (DGGS) [1], [9]. A DGGS encodes a globe as a hierarchy of discrete indexed cells, each representing an area on the surface of the Earth, in contrast to conventional geographical coordinate systems that encode infinitesimal point locations.

Latitude/longitude discretizations exist, but are problematic due to the convergence of lines of longitude at the poles. The resulting cells vary immensely in area causing, among other challenges, difficulties in sampling and determining the appropriate level of detail when requesting data. Ideally, in a DGGS, the cells at a particular level of the hierarchy should all be of approximately equal area, otherwise challenges arise in sampling and in determining an appropriate resolution at which to request and display data.

One method for achieving consistent cell areas is to initially approximate the globe with a polyhedron, such as a cube or icosahedron. The faces of this polyhedron represent the base-level of cells for the DGGS. A hierarchy is then constructed by repeatedly subdividing the cells down to an arbitrary level of detail and projecting them to the surface of the globe. There are subdivision schemes that produce children that are regular and of equal area to their siblings [4].

Many different strategies exist for the creation of a DGGS, differing by cell shape, initial polyhedron, cell refinement and indexing schemes, and the type of projection. These aspects allow for many different variations in the implementation of a specific DGGS, each with its own possible strengths and weaknesses. For a more in-depth review of DGGS, Mahdavi-Amiri et al. [4] survey existing DGGSs and their uses.

As only three cell shapes – triangles, quadrilaterals, or hexagons – can be tiled regularly, most DGGSs employ these cell shapes [4], [9]. Using hexagonal cells produces superior sampling and a uniform neighborhood [5], [6], but quad cells are a better choice when defining hierarchy or rendering [10], [11], [12]. Due to their desirable properties, hexagonal cells appear in several DGGS implementations [13]. Hexagons are also easily sampled [14], [15], and a simple grid conversion to the other cell types [7], [8] allows us to bypass the lack of direct hardware support for hexagons as well as the difficulties in hexagonal indexing and refinement.

Various types of initial approximating polyhedrons have been used in different DGGSs. Typical choices include tetrahedrons [16], cubes [17], [18], [19], octahedrons [20], [21], [22], dodecahedrons [23], and truncated icosahedrons [6], [24], [25].

To represent finer level of details, smaller cells need to be produced. Apart from being very similar in shape and area, cells of a particular detail level also need to be produced in a predictable order such that the DGGS cells remain able to be indexed; Mahdavi-Amiri et al. [8] present various indexing methods and conversions between them. This index is used to store or retrieve a unit of data from a DGGS and is analogous to a latitude/longitude coordinate. Refinement methods include 1-to-4 refinements for triangular cells [21], [26], [27], 1-to-2 and 1-to-4 refinements for quadrilateral cells [17], [28], and 1-to-3 [6], [29] and 1-to-4 refinements for hexagonal cells [30], [31], [32]. The ability to access data at predictable detail level, size, and location is a powerful functionality that is essential to our work.

The DGGS also needs to account for the distortion produced when the cells are projected onto the globe from their planar parents. Equal area projection methods are usually preferred for the purpose of data integration, and are used extensively in cartography [33], [34], [35].

2.2 Multilevel Focus+Context Visualization

Conventional Digital Earths suffer from visualization difficulties stemming from the scales and distances involved in geospatial data. It is difficult to visualize regions that are vastly different in scale as a single perspective is often insufficient. Viewing two or more distant areas at a high level of magnification is also problematic with a single point of view. Multilevel focus+context visualization techniques overcome this limitation by providing multiple dynamic views of the data at customizable resolutions and perspectives.

In the fields of information and scientific visualization, Hauser generalized the common methods for discrimination between focus and context based on their degrees of importance [36]. The use of a virtual lens to magnify foci is a common method for discriminating between focus and context [37], [38], [39]. Such visualizations draw their inspiration from preexisting techniques common in medical and scientific imagery, where the artist highlights areas of interest with magnified sub-images [40]. Our implementation uses a discontinuous undistorted technique [37], rather than the continuous style [38], [39], as the views are discrete and independent.

Packer et al. [41], [42] created a system allowing for multilevel focus+context visualization of layered tube-shaped structures with continuous artistic transitions between different levels of details. However, their work does not allow branching structures, unlike ours. Other works include multifocal visualization of medical data by Ropinski et al. [43]; a multifocal, multilevel system by Cossalter et al. [44] for visualizing networks; a multifocal treemap visualization by Tu and Shen [45]; and multifocal and multicontext augmented reality systems by Mendez et al. [46] and Kalkofen et al. [47], respectively. Our approach differs from these works in that it is based on view-dependent resource allocation derived from the viewing volume of the foci.

The work of Hasan et al. [48], [49] produced a system allowing the exploration of large-scale 2D and 3D image data through a multilevel focus+context environment. The work is based upon a balanced wavelet transform [50], and produces a tree structure where each level of magnification is available for further magnification by adding additional lenses. Except for the root, each node in the tree can represent both focus and context in this environment and has varying DOIs [3] directing resource allocations. This was a primary motivation for our work, in which we have added virtual cameras to our tree nodes.

3 Methodology

Our methodology is broken down into topics covering how we reference data, construct the view-aware globe, and build the data visualization system.

3.1 System Overview

As illustrated in the visual synopsis of our system in Fig. 2, tasks are divided between a server and one or more clients. Each client machine runs our client-side web application within a web browser, accessible via a URL. The web application houses a DGGS with a quadrilateral-based hierarchy of cells. We refer to this as the client DGGS. A client is responsible for determining which cells (based on each's visibility within the current views and each's resulting screen size) need to be downloaded. A client sends queries for the desired data to the server and stores the query responses within its simplified DGGS.

The server has access to a database of geospatial datasets, referenced using a DGGS. When this server is queried with a cell and a dataset, it samples the DGGS-referenced data and produces a quadrilateral result for the query via grid conversion. The query result takes one of two forms: the geometry of the cell being requested or a data texture representing the data thereupon.

3.2 Data Accessibility

In this subsection, we explain the details of the DGGSs in use within our system. We also demonstrate an adaptation of DGGSs into a web-based client-server model.

3.2.1 Discrete Global Grid System

A core concept in DGGS is the idea that a cell with a particular index represents a physical piece of the Earth and is a bucket to which geospatial data may be assigned. This cell has a particular resolution, position, and size. Every dataset within the DGGS has an identical cell structure. This feature allows for multiple datasets to be combined without the need to track any per-dataset information such as position or scale; this information is intrinsic to the grid. This stands in contrast to, e.g., raw satellite imagery, in which samples are typically aligned with the path in which the satellite is travelling. It is difficult to render or analyze this data as each sample has its own position and scale. This problem compounds once other datasets are combined, each potentially with its own reference system. By using a DGGS, we bypass these difficulties via a universal grid system that does not vary from one dataset to another.

Though DGGSs often include complex indexing and projection schemes, these are not needed on the client-side if the server has the ability to serve cell geometry. The two tests that the client needs to perform are tests for on-screen visibility and cell size, both of which may be performed on this geometry. This approach enables us to support interoperability with different DGGSs, since the client is not dependent upon a particular DGGS and can interact with a variety of data providers' servers without modification.

The client requests geometry and data starting at the coarsest level of resolution, and in regions of interest (ROIs) it requests refined cells until an appropriate level of detail is acquired. This requires the client to be able to predict the indices of children cells, which is easily provided by the client-side hierarchy.

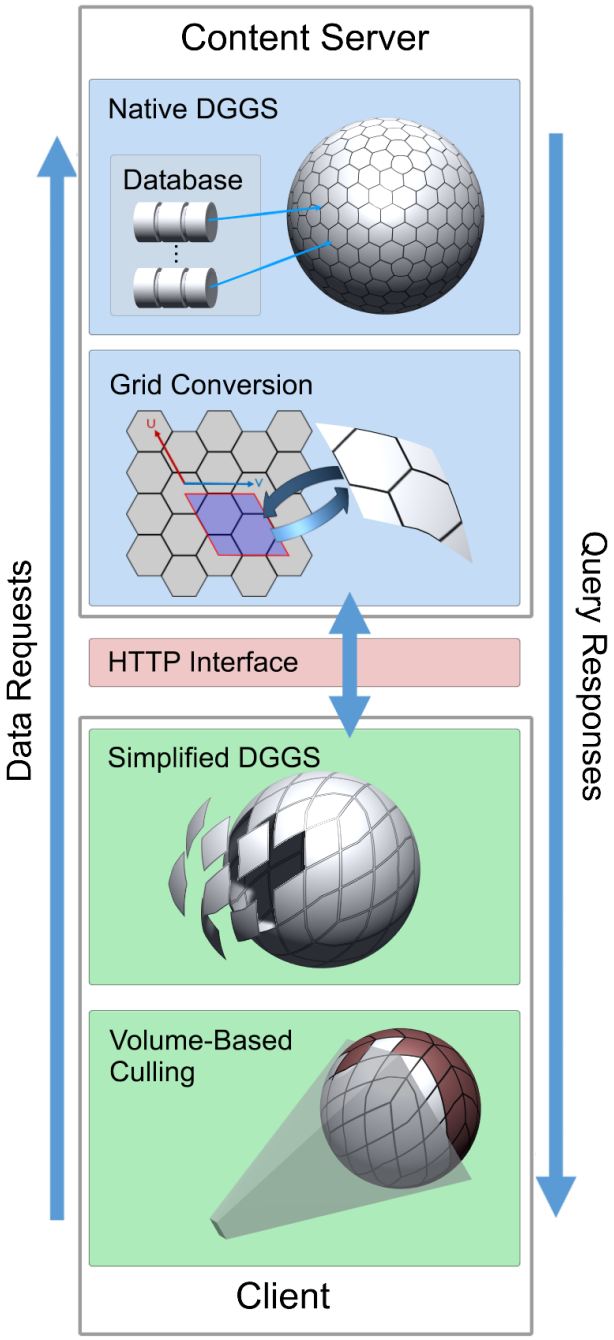


Fig. 2. High-level overview of the system.

3.2.2 Client-Server DGGS

It is impractical to load full datasets onto the client due to limitations of memory and network throughput. To reduce the amount of data transmitted between the client and the server, we limit the data to only that which is required to render the globe from the current views. As views are changed (e.g., by zooming or panning) or added, the data needed to accommodate these views are requested and downloaded. The client requests the missing data from the DGGS server as shown in Fig. 2. Since the exact size and resolution of cells are explicitly available, a DGGS provides a useful framework for querying, and allows the client to be very specific about which data are needed to complete a

view.

In our client-server framework, the server transmits two types of information for rendering geospatial data on the client. The first type is globe geometry, consisting of 3D meshes that approximate the curved surface of the globe within a chosen cell. As DGGS cells are indexed at various levels of resolution, this system can be queried in a manner capable of producing the mesh for a globe, in part or in whole, at a wide variety of resolutions. The second type of transmitted information is the cell-based data that are to be visualized on the globe. To acquire these data, the client query references a data-source and a cell, and the server responds with a sampling of data values within the cell. Since the cells in both of these query types represent the same areas on the globe, their correlation is straightforward.

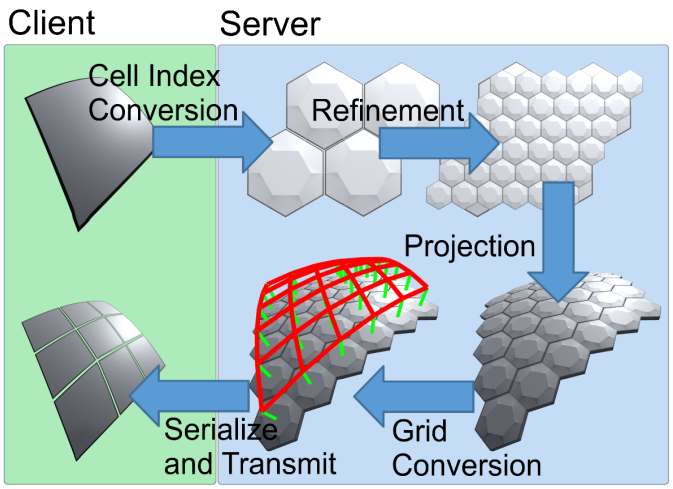


Fig. 3. An example Client-Server cell refinement process.

To reduce the processing load on the server, tasks such as rendering the globe and visualizing requested data is done using the CPU and GPU resources of the client. Additionally, for the sake of efficiency, our server behaves in a stateless manner. This means that it does not track the individual requirements of each attached client. Therefore, the client is also tasked with monitoring its downloaded cells and for querying any missing data at its own discretion. To accomplish this, the client maintains a tree structure containing its downloaded cells.

The DGGS underlying the server and database is typically not tuned for rendering or data transmission in a web-based environment. As shown in in Fig. 2, the native DGGS [5] on our server is built using hexagonal grids. Hexagons have numerous advantages in the context of a DGGS, but require specialized methods for transmission and are not natively supported by graphics hardware.

To address this issue, the server converts the data into a format compatible with the client DGGS via efficient hierarchical grid conversion [7]. This conversion technique allows us to pack and convert hierarchical hexagonal grids into hierarchical quadrilateral grids with their own indexing methods. This packing can be seen in Fig. 3. The resulting indexing method allows us to build a tree of downloaded cells on the client-side, which can be efficiently traversed. Moreover, the resulting quadrilateral cells are compatible with the rendering pipeline, including shaders' 2D texture samplers.

DGGSs use a variety of projection schemes for to map planar data to the surface of the Earth. To maintain interoperability with different DGGSs, our system performs the projection on the server. This frees the client of the need to consider the often complex projection formulas specific to any particular server. A result of this, however, is the client cell hierarchy may not be perfectly congruent. To correct for this, we use bounding volumes that are slightly larger than the cells (such that the bounding volume is guaranteed to contain all descendant cells) when performing on-screen visibility tests. This prevents false-negatives in these tests, which would result in missing geometry near the edge of the screen.

Though the discrete nature of a DGGS makes it immediately applicable to raster data, DGGSs are capable of working with vector data as well. Vector data are common in datasets related to boundaries or areas (for example, a dataset containing all water bodies in Canada), and datasets can be very detailed due to the complexity of the boundaries of natural landmarks. This complexity is addressed by our server-side DGGS through the conversion of vector data to cell-based rasters for transmission to and rendering on the client. Furthermore, the multiresolution hierarchy of our server-side DGGS enables a level-of-detail representation of the vector data aligned with the underlying grids at various resolutions.

3.3 View-Aware Globe and Multiple Views

Our main approach to multilevel focus+context visualization is to utilize multiple virtual cameras. Each camera is responsible for the live creation of one view of the Digital Earth. In this section, we detail how our system manages appropriate levels of detail and determines what new data to download. We also show how our system allows multiple simultaneous views of the globe and how this enables multilevel focus+context visualization.

3.3.1 Spatial Decision Making

To render a high-detail DE in real-time, a system has to be discerning about its allocation of resources. A complete high-detail globe is impractical to download or render in real-time. Instead, our client-side globe tracks the present cameras and adapts to each of them. Such adaptation ensures that the portions of the globe that are rendered are those that can be seen by the camera and are of appropriate resolution. Therefore, we base our method on two factors – determining what parts of the globe are visible to the camera, and determining the appropriate level of detail that should be presented to a camera.

We determine which cells are outside of the view camera frustum so they can be ignored by the rendering process. The tree hierarchy of the client DGGS, introduced in Section 3.2, is useful for determining the visibility and scale of portions of the globe. Additionally, to determine the appropriate level of detail, we determine the ratio of visible data samples in a cell to the number of screen pixels that it covers. For each cell (representing a quadrilateral region on the surface of the Earth) that is stored in the tree, a bounding volume

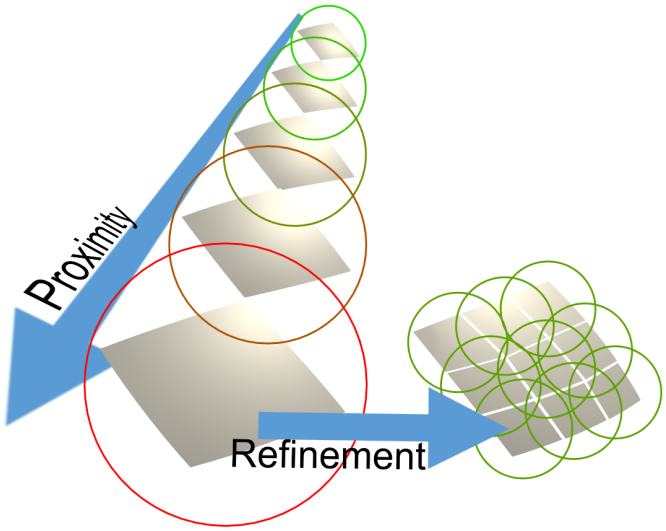


Fig. 4. Bounding volumes used as an indicator for globe refinement.

is calculated. In our implementation, we used bounding spheres, though other types of volumes are possible.

The bounding volumes are used alongside a camera to calculate the approximate size of a cell on the screen. By projecting the bounding volume into screen space via the camera, it is measured in pixels. By measuring the approximate ratio of data samples (i.e., texels in a data texture) in the cell to the number of screen pixels the cell occupies, we calculate a factor that determines whether the cell's resolution is appropriate for the current view. If this ratio is lower than a certain threshold, the cell is too large on the screen and is then replaced with its children. On the other hand, if this ratio is high, the cell is too small and contains data that are sub-pixel on the screen. In this case, its rendering costs exceed its benefit to the visualization, so it (and its siblings) are replaced with the parent cell instead. Fig. 4 shows cell bounding volumes (represented by circles for simplicity), and the refinement of a single cell as it is replaced by its higher-resolution children.

The bounding volumes are also used to determine whether a particular cell is visible to the camera. If the bounding volume of a cell does not intersect the viewing frustum, the cell and its children are not visible. This property makes the culling of the globe efficient as entire branches of the tree can be quickly pruned without the need to test every individual cell in each branch.

This culling process can run many times between frames of animation. This allows the globe to adapt to a multitude of concurrent cameras. We render the cameras sequentially, adapting the globe to each in turn just before it is rendered.

3.3.2 Globe Tree Traversal

In our method, the processes of rendering the globe and queuing the download of missing data are combined into a single tree traversal.

Since our client starts with very little information about the DGGS it is about to use, it requires a starting set of data. Therefore, when first created, the client requests from the server a list of indices for the cells at the coarsest resolution and queues these cells for download. Once the geometry for these cells are received, it becomes possible to determine each's size and location, and to then traverse the client's DGGS tree for each camera using Algorithm 1.

Algorithm 1: Tree traversal for requesting data and culling.

```
input: node, a cell in the client DGGS
if node is not in the camera frustum then
    return
end
if data texels to screen pixels ratio for node is sufficiently large then
    add this node to the render queue;
else
    foreach child node do
        if child node has no data then
            queue data download for child node;
        end
    end
    if any child node had no data then
        add this node to the render queue;
        return
    end
    foreach child node do
        call Algorithm 1 recursively;
    end
    if no children were added to the render queue then
        add this node to the render queue;
    end
end
```

Algorithm 1 recursively produces a list of cells that should be rendered for a particular camera for a single frame, while it queues the download of missing data for cells. Since these downloads are an automatic by-product of our visibility checks, any camera in the scene can trigger the globe to download additional data as needed.

3.3.3 Multiple Views and Multilevel Focus+Context

While, by this point, we have all the tools needed to create multilevel focus+context, the visualization itself is still missing. Our DE can be explored at a variety of scales and with full mobility for the virtual cameras. This produces challenges for how the foci and contexts fit within the scene and how they are rendered. Our system supports dynamic multilevel focus+context visualization, in which multiple ROIs can be viewed simultaneously, regardless of proximity or difference in scale. These different views form a multilevel, multiview hierarchy and provide context cues where applicable, allowing a viewer to easily identify the areas or different types of data being emphasized.

We adapt the method proposed by Hasan et al. [49] for the creation and management of our multilevel focus+context visualization hierarchy. Initially, a new ROI can be chosen on the globe view interactively. This may be achieved in a variety of ways, such as by sketching strokes or by drawing a bounding box. A new camera is then created in the scene so that its viewable area matches the chosen ROI. Next, a magnified view of the ROI as viewed by the new camera

is rendered separately on the screen. Finally, we render semitransparent connections between the chosen ROI and its magnified view. In the focus+context creation scenario thus described, the ROI is located on the initial globe view, which provides contextual information. Additionally, however, a magnified view of an ROI can recursively serve as the context for new ROIs. This allows the system to facilitate the creation of a multilevel focus+context visualization hierarchy (see Fig. 1, for example).

Each of the newly created cameras is dynamic and may move about the globe or change zoom level as desired. Alternatively, these cameras may be moved or animated procedurally. For example, a camera can be made to follow the orbit of a satellite, providing a live view from the satellite's point of view.

As discussed in Section 3.3.2, our globe adapts to the views of cameras, showing the globe from each view in the scene an appropriate level of detail. Because downloading new data is also a byproduct of this process, creating, moving, or zooming any of these cameras requests missing data where applicable. As this process only downloads missing data, regions viewed by two cameras at the same level of detail do not require the data to be acquired twice.

3.4 Client-Customizable Visualization

One of the goals of this work is to provide a toolset for diverse visualization and styling of data on the globe. As a consequence, our system also reduces the amount of work performed on the content server and improves web caching performance. This section explains how we encode data on the server-side for transmission to the client and our client-side data-rendering process.

3.4.1 Texture Encoding

Digital globes, such as Google maps, often use a secondary layer to overlay information or additional imagery on the surface of the sphere. The tactic of replacing the surface texture of the globe works for a single overlaid dataset but starts to create problems as additional datasets are added. When a dataset is converted to a texture, the underlying values are often obfuscated by the styling. Combining these datasets for visualization is then difficult because multiple datasets might use the same colour or pattern for multiple purposes. We overcome this problem by encoding the layers of our globe in a different manner. Since web browsers and shaders can efficiently work with images, we encode the various datasets as images. Rather than producing an RGB image to be rendered directly on the globe, the server converts data layers to multiple raster images (RGBA channels). These raster images are packed as a series of textures, called data textures (as shown in Fig. 5.) that are then transmitted to the client. On the client-side, they are used to create various visualization styles of the input datasets.

Up to four data layers can be encoded in standard RGBA image file formats such as PNG. Image files compress well and are compatible with the content caching mechanisms that are in common use on the internet, in addition to our own server-based caching strategies. Since the colour channels in an image are only one byte each per pixel, these channels

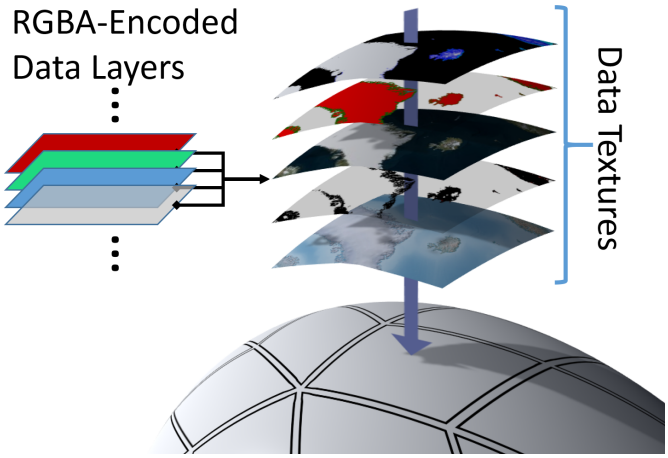


Fig. 5. From data layers to data textures.

are not capable of containing all possible value ranges. If the range of the data values is $[\alpha, \beta]$, we need to map this range to [0, 255] and quantize to integer bins. For many visualization tasks, this range is sufficient. However, in some cases, the loss of data granularity would be disruptive. In such cases it is possible to use more than one channel for the data. For a full 32-bit (four-channel) image, 4.2 billion values are available, providing sufficient precision for many visualization scenarios.

A typical GPU in the WebGL environment has access to 16 texture units, meaning that in a single render pass 16 individual textures may be sampled. With up to four data layers packed into a single texture, this allows 64 possible data layers, sufficient for many practical applications. In a rare situation, if more layers of data need to be visualized, multiple passes of transmission and rendering can be used.

3.4.2 Data Styling

To meet the requirement that our globe should be capable of diverse styles, we introduce a novel approach to texturing. Styling could be done exclusively on the server-side, however this lessens the benefit of web caching and increases the work needed to be done by the server. In such a scenario, if two clients request identical data yet under different styles, the server will have to work twice for every tile visible to both clients. Any time the style is changed on the client-side, the request needs to be handled by the server yet again.

To allow a better solution to this problem, we perform styling on the client-side. To even better leverage the client's computational power, we use the client's GPU to process styling. A given cell on the globe can have several datasets visible on it at a time. For our quadrilateral cells, the data are sampled and stored in a 2D array. When fed into the shader, these values are rendered into colours and mapped onto the surface of the globe.

Imagery and vector datasets associated with the globe are commonly rendered as RGB values. Instead of treating the image channels at colour values, our system provides further styling options by allowing the mapping of input data to attributes such as emission, height, glossiness, or even animated patterns. Since information can be conveyed by variations in patterning or glossiness, for example, this

expanded styling toolset increases the amount of data that can be layered on the same area of the globe.

The most direct approach we use is to employ a styling matrix, M_{style} and calculate

$$P_{lighting} = T_{data} M_{style} \quad (1)$$

where T_{data} is the data texture and $P_{lighting}$ are the lighting parameters (e.g., red diffuse, glossiness, emission, etc). The styling matrix can be easily modified on the client to produce a wide variety of combinations and styling of the data. A limitation here is that M_{style} can only produce a linear remapping of the data values. However, shaders can be assembled from discrete modules and compiled very quickly, so a more general solution is also possible,

$$P_{lighting} = f(T_{data}) \quad (2)$$

where f is an arbitrary mapping function. This f is any mapping function, written as a snippet of shader code.

Styling functions can be changed simply by changing shader uniforms, or by replacing or modifying the shader program. Because such modifications are inexpensive, they can be used in interesting ways. In our implementation, we experimented with *per-view styling*. Instead of attaching the styling matrix, M_{style}, to the globe, it is attached to each camera. This permits each camera to have its own, unique view of the data. This technique can be used to emphasize certain data on zoomed-in views, or even show entirely different data in different views by suppressing datasets in the styling. An example use case is opening two magnified views of the same city – one view showing streets and neighbourhoods and the other showing population density. A third view could be used to show a blending or combination of the two.

4 Performance Considerations

The performance of the proposed system is a major factor in its design, as there are various interacting constraints. For our solution to work in practice, the server should be efficient, and ideally should benefit from technologies that scale well to large numbers of clients. The client's limitations also needs to be considered, as caching large amounts of data within a web browser is not practical; only the RAM of the client machine should be considered available. Our system achieves a balance between these constraints while providing benefits to rendering and interactivity over current technologies. Here, we examine the performance characteristics of the main areas where performance bottlenecks could occur. These areas are the tree culling algorithm, which needs to be efficient in order to support multiple views, and the data encoding and caching mechanisms. These factors impact memory, network, and CPU use on both the client and server.

4.1 Tree Hierarchy and Culling

Fast and appropriate culling is vital to this work. In order to render multiple simultaneous live views of a globe in real-time, we need to produce visible geometry that is suitable for fast rendering. Additionally, this decision needs to be quick as it has to be made for each camera in the scene. We expect our spatial tree to be able to cull data very quickly. We validated this hypothesis experimentally using Algorithm 1. For instance, we tested a render made up of cells predominantly at the 8th resolution. At this resolution, it would take 430,467,210 quadrilaterals to fully cover the globe. As shown by the examples in Table 1, our system only needs to consider 495 of these quadrilaterals in the process of determining the 67 required for this rendering.

TABLE 1. Comparison between the number of cells checked for visibility, cells visible to a camera, and the total number of DGGS cells at various resolutions.

Level of Resolution	Checked Cells	Visible Cells	Cells at given Resolution
1	90	45	90
2	153	78	810
3	279	41	7290
4	315	34	65610
5	351	33	590490
6	414	56	5314410
7	486	56	47829690
8	495	67	430467210

A drawback to using downloaded globe geometry to populate a spatial tree is that lower resolution regions must be downloaded before their higher resolution children. However, the overhead of a single parent cell is shared by all of its children. Thus, on average, the overhead is $\sum_{r=1}^{R} \frac{1}{n^r}$, where n is the number of children cells in the multiresolution scheme and R is the level of resolution for a viewed area. Experimentally, we find that this is close to the trend in our system. From the example above, the 67 visible quadrilaterals required only 7 cells from coarser resolutions to support them within the hierarchy, close to the 7.44 we would expect.

4.2 Data Transmission and Encoding

There are different approaches for encoding the various data that appear on a globe. For our visualizations, we mainly focused on raster and vector datasets, such as digital elevation models and political boundaries.

As discussed in Section 3.4.1, we rasterize all types of datasets. This is justifiable for vector datasets due to their potential complexity: commonly used datasets such as political boundaries may in fact be comprised of hundreds of thousands or millions of data points. Rendering this kind of dataset in real-time is a challenging task. Therefore, vector datasets are converted to discrete data within a DGGS and rendered to a texture, allowing a sampled approximation of all these boundaries to be produced and rendered efficiently. Our system treats these sampled vector data identically to raster data and therefore incurs no overhead from vector datasets.

Our primary method of encoding raster data is in the form of square textures which are overlaid on quadrilateral regions on the globe. This texturing system uses the indexing scheme of Mahdavi-Amiri et al. [7], [8], which can produce square textures wherein each texel represents a single cell of the DGGS from which it was sampled. In essence, these textures can be viewed as a method of batching a series of cells for transmission. With this in mind, next we compare this process against the transmission of the hierarchical DGGS data if it were transmitted in some other format.

If the data were, for example, batched as a hierarchy of hexagons rather than data textures corresponding to the quadrilateral regions used in our client-side DGGS, the total amount of data sent would be similar. The hierarchy would, however, represent a region that is difficult to access within a shader. It would be possible to run the grid conversion algorithm [7], [8] on the client to take some load off the server. We considered this model, but opted for the server-side grid conversion largely due to the asymmetry in the environments between client and server. The server has the ability to work with large hard drives and sizeable caches, whereas a browser-based client is largely limited to just RAM. This allows the server to access small sections of a large dataset to handle the grid conversions per-request. A client-side grid conversion would likely require the client to maintain a memory-resident cache of cellular DGGS data as well as an entirely duplicate set of render-friendly rectangular textures and geometry.

Our method of packing datasets into data textures is not exclusively used for improving the visual results; it also produces some considerable performance benefits. A particular dataset can be used for different purposes by different users. On one visualization, an elevation dataset may be used for shading terrain, whereas on another it could be used to colour a topographic map. To meet the demands of both clients, the server only needs to generate a single texture. This result can be used by both clients for their respective visualizations. This considerably improves the server's efficiency as it is able to cache results from a wide variety of visualizations with a relatively small number of textures.

Similar benefits exist on the client-side as well. Some data textures on the globe can be reused between visualizations. If the client switches between two renderings where geopolitical boundaries are visible, the textures defining these boundaries do not need to be re-downloaded, even if they are styled differently. Our interactive data styling method utilizes this property. Since styling is purely a choice on how the textures are converted into colours on the screen, no new request for data is sent to the server when styling parameters are altered.

Though web caching works well with our system, there is a second type of caching that additionally helps with the overall server performance. A web cache will only get a successful cache hit on a texture if all four of the image channels match a previous request and if they were normalized and quantized with the same functions. Cache hits are therefore most probable if the images contain data which are commonly used together. To improve cache efficiency further, we also cache the individual data layers before they are packed into data textures for transmission to clients. This allows for cache hits even when a client asks for data arranged in a layer order not previously requested.

5 Results

Since an interface built atop a DGGS provides access to many datasets, we have been able to produce a variety of results using our implementation. Each render in this article is the product of multiple datasets used in a variety of manners.

In Fig. 6 we show examples of our data styling shaders, and how they are influenced by the lighting of the scene in which the globe resides. Fig. 6(a) mixes an artistic colouration of the globe with a digital elevation dataset that is used to produce shading and normal information for the purpose of enhancing topographic and bathymetric features. In Fig. 6(b), the textures for topographic and bathymetric shading are reused from Fig. 6(a) and do not need to be re-downloaded if the client transitions from one to the other. This globe uses Bing maps for a base texture, while a vector dataset of world political boundaries is mapped to luminosity and height to increase visibility. Fig. 6(c) uses a combination of world political boundaries and world population densities. In this render, the population density data is mapped both to a colour as well as luminosity, allowing populated places to glow in the absence of light. Fig. 6(d) shows the interaction of light with datasets when luminosity is used.

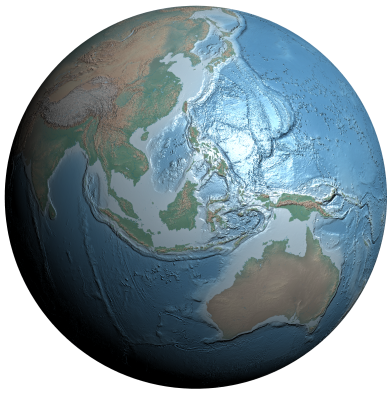
Fig. 6(e) shows the versatility of styling matrices. The three types visualization shown use the same four underlying data textures on the globe. Styling matrices can be used to produce a variety of visualizations, emphasizing different aspects of the data. These styling matrices may be blended or altered in real-time on the client, producing smooth animation from one style to another. Since these styling calculations are run on the client, these changes are made without any additional interaction with the server.

Providing an interface that can map data to various lighting parameters allows us to produce a variety of complex results. Figures 6(f), 6(g), and 6(h) show globes which use physically-based shading to produce materials that react realistically to their environments. The ability to integrate physically-based shading is important for producing 3D scenes with consistent lighting throughout, and is useful as it allows for digital globes to be used in a wider variety of applications. Rather than being confined to a purpose-built application designed to render only the Earth, the globe can be integrated seamlessly into larger 3D scenes.

Fig. 7 shows a collection of multilevel focus+context visualizations possible with our system. Fig. 7(a) illustrates one of the primary benefits of this type of visualization on a virtual globe. Within a more typical system, it would be difficult to have a side-by-side comparison of regions that are not nearby. Our system facilitates easy and effective comparisons of ROIs in very distant locations or at very different scales. Despite the scales and distances involved, the respective contexts allow a user to easily perceive scale and location of the data being viewed.

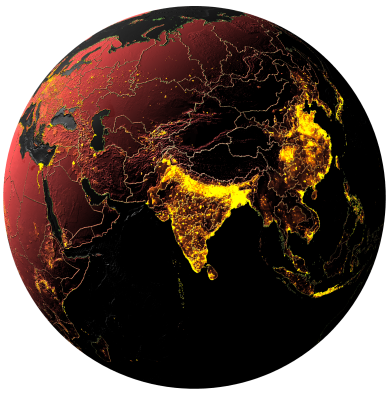
Next, Fig. 7(b) highlights the geographical awareness that is made possible with multilevel focus+context visualization. This single visualization conveys the location of the University of Calgary within a view showing an entire continent. In Fig. 7(c), we display the population densities of a few ROIs on the Earth. In a single-view DE visualization, it would not be possible to show this level of detail for areas so far apart.

Finally in Fig. 7(d), we illustrate another benefit of a client-side rendering of data. Here we present a flood warning dataset, which originally comes in the form of vectors. When overlaying such a dataset on a map, the result can be confusing because the colours used by the dataset may conflict with those employed by the map. To avoid such confusion in this visualization, we apply an animated water-

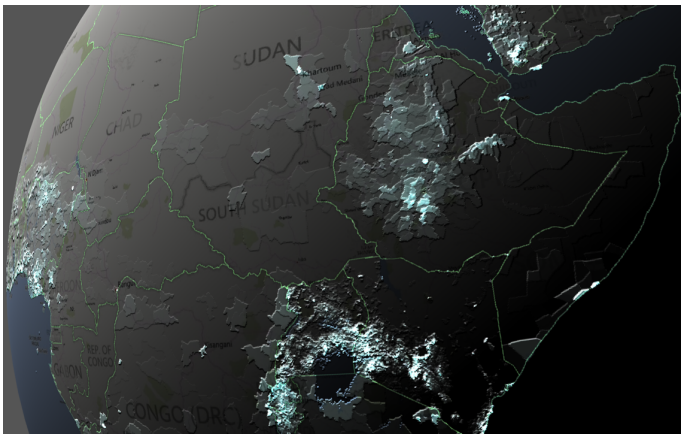


(a) Artistic texture and shading for topography and bathymetry (S1, S3).

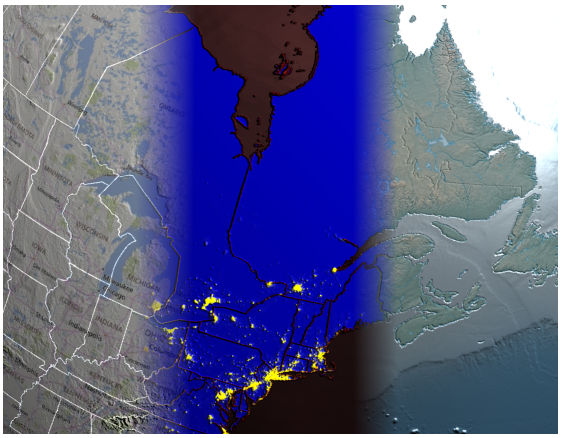
(b) Map textures and embossed political boundaries (S1, S2, S4).



(c) Population data and political boundaries as luminous textures (S1, S2, S5).



(d) Population density superimposed on map data (S2, S4, S5).



(e) Three different visualizations of the same data but using different styling matrices (S1, S2, S3, S4, S5).

(f) Glossy material (S1, S3).

(g) Plastic material (S1, S3).

(h) Metallic material (S1).

Fig. 6. Various client-side data styling scenarios.

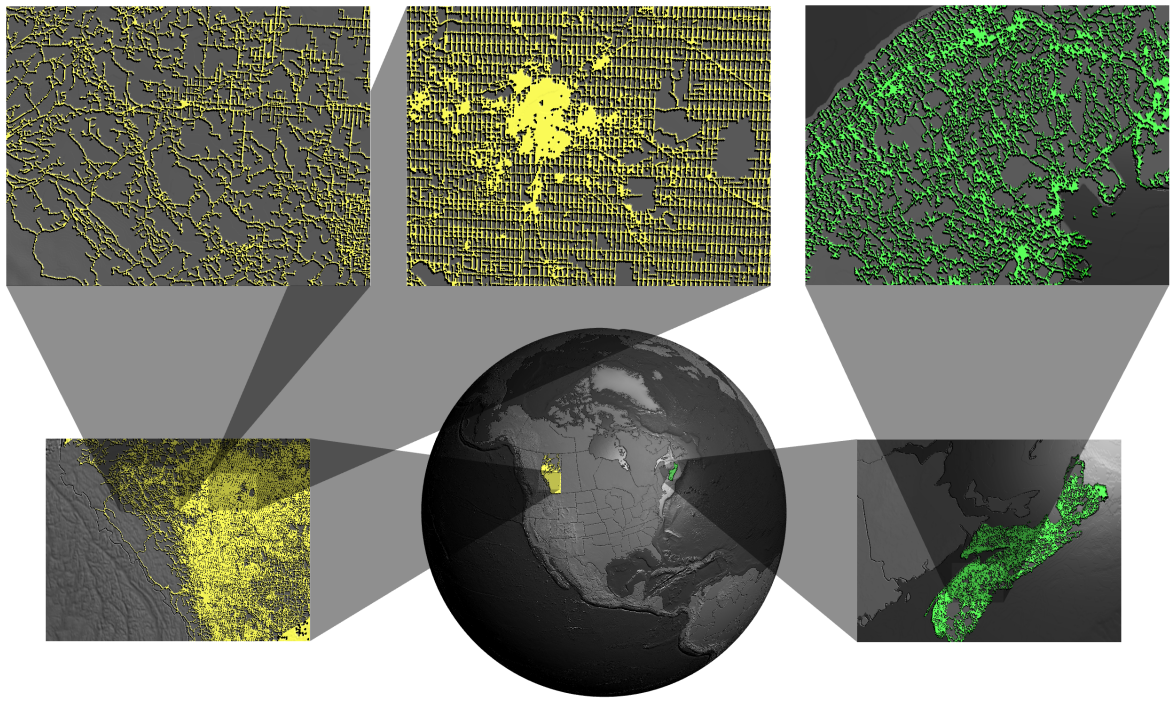
like ripple to the flood dataset, allowing it to stand out strongly against the background map. As all of our views are real-time, these regions animate within the various foci and on the base globe simultaneously.

See our developed prototype in action at 1x speed in the supplemental video available from the publisher's website.

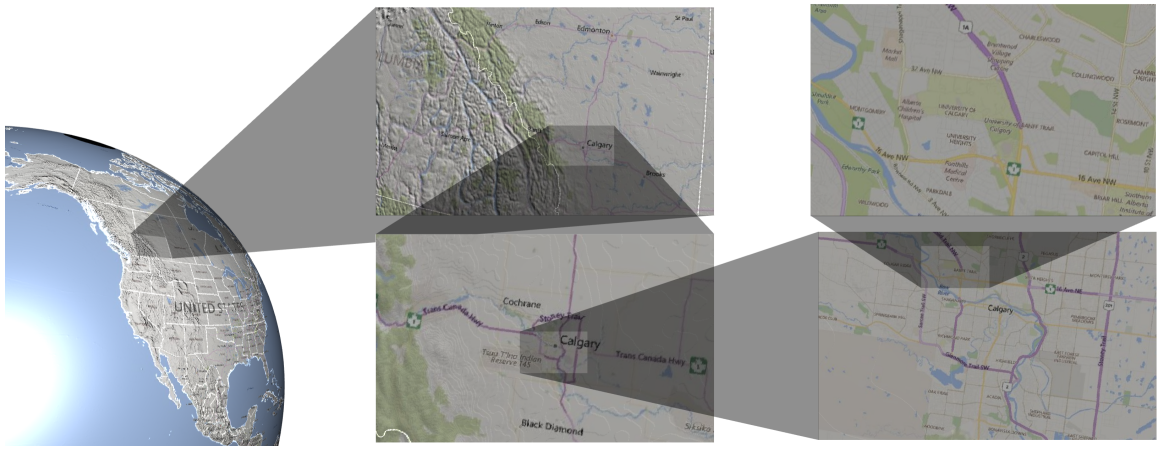
6 Conclusion and Future Work

Our approaches to DE visualizations overcome several major hurdles in the visualization of geospatial data. Our system also allows our globe to adapt to a particular view and render in real-time, enabling a multitude of concurrent live views to be produced for every frame of animation. Using this, we overcome screen space limitations through multilevel focus+context visualization, applied interactively on our globe. We also demonstrated the integration of disparate data sources through the use of DGGS.

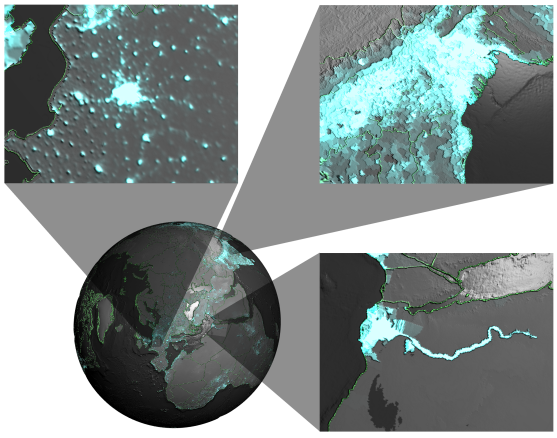
We leveraged the ability of DGGS to correlate data layers to provide a styling system that is able to produce a wide variety of visualizations. By separating the styling from the data textures, we provide a mechanism for improved server caching, and thereby improve our system's scalability.



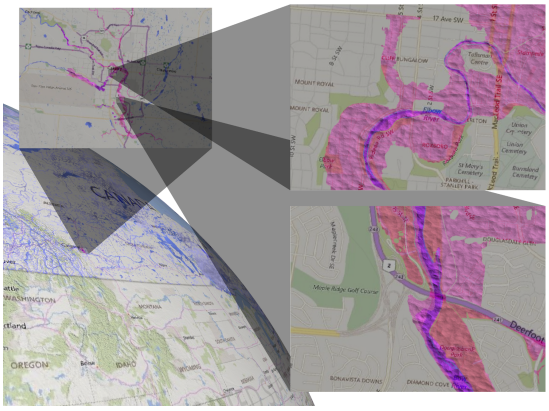
(a) Comparison of Canadian road networks in Alberta and Nova Scotia (S1, S2, S7, S8).



(b) Diverse levels of magnification (S1, S2, S4).



(c) Population densities in different parts of the world (S1, S2, S5).



(d) Calgary water bodies and flood projection (S4, S9, S10).

Fig. 7. Various focus+context visualization scenarios.

Relevant future works include the expansion of this system to handle large-scale time-varying data. An example of this would be smoothly and interactively traversing through years of Landsat data. We are also interested in the exploration of data-driven procedural content on the surface of the globe, such as the generation of 3D forests with high levels of detail. Additionally, we would like to expand the set of data styling functions for use on the client side.

Acknowledgments

We would like to thank Idan Shatz and Ali Mahdavi-Amiri for their insightful discussions and Troy Alderson for his editorial comments. This research was supported in part by the National Science and Engineering Research Council (NSERC) of Canada and *the PYXIS innovation inc.*

To produce the images used in this paper, our system used a variety of datasets, listed below. Licenses were acquired for those datasets not freely available for use.

(S1) NOAA, 2-Minute Global Relief, www.ngdc.noaa.gov
(S2) ESRI, World Political Boundaries, www.arcgis.com
(S3) Natural Earth, www.naturalearthdata.com
(S4) Microsoft, Bing Maps, www.bing.com/maps
(S5) Columbia University, Gridded Population of the World, sedac.ciesin.columbia.edu
(S6) NASA, Land Surface Temperature, neo.sci.gsfc.nasa.gov
(S7) Highway Geomatics Section (HGS)/Alberta Transportation, Alberta Road Network, geogratis.gc.ca
(S8) Government of Canada/Natural Resources Canada, Nova Scotia Road Network, geogratis.gc.ca
(S9) Statistics Canada, Canadian Inland Lakes and Rivers, statscan.gc.ca
(S10) City of Calgary, 1:100 Year Flood Map, www.calgary.ca

References

[1] M. F. Goodchild, "Discrete global grids for digital Earth," in *Proc. of the 1st International Conference on Discrete Global Grids*, 2000, pp. 1–9.
[2] A. Gore, *Earth in the Balance: Ecology and the Human Spirit*. Emmaus, PA, USA: Rodale Press, 1992.
[3] S. K. Card and D. Nation, "Degree-of-interest trees: A component of an attention-reactive user interface," in *Proc. of the Working Conference on Advanced Visual Interfaces*. New York, USA: ACM, 2002, pp. 231–245.
[4] A. Mahdavi-Amiri, T. Alderson, and F. Samavati, "A survey of digital earth," *Comput. Graph.*, vol. 53, Part B, pp. 95–117, 2015.
[5] the PYXIS innovation inc., "How PYXIS works," https://goo.gl/4U9Xz1, 2015.
[6] K. Sahr, "Location coding on icosahedral aperture 3 hexagon discrete global grids," *Comput. Environ. Urban Syst.*, vol. 32, no. 3, pp. 174–187, 2008.
[7] A. Mahdavi-Amiri, E. Harrison, and F. Samavati, "Hierarchical grid conversion," *Comput. Aided Des.*, vol. 79, pp. 12–26, 2016.
[8] A. M. Amiri, F. Samavati, and P. Peterson, "Categorization and conversions for indexing methods of discrete global grid systems," *ISPRS Int. J. Geoinf.*, vol. 4, no. 1, pp. 320–336, 2015.
[9] K. Sahr, D. White, and A. J. Kimerling, "Geodesic discrete global grid systems," *Cartogr. Geogr. Inf. Sci.*, vol. 30, no. 2, pp. 121–134, 2003.
[10] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Planet-sized batched dynamic adaptive meshes (p-bdam)," in *IEEE Visualization*, Oct 2003, pp. 147–154.
[11] N. Greene, "Environment mapping and other applications of world projections," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 21–29, 1986.
[12] K. Compton, J. Grieve, E. Goldman, O. Quigley, C. Stratton, E. Todd, and A. Willmott, "Creating spherical worlds," in *SIGGRAPH Sketches*. New York, USA: ACM, 2007.
[13] K. Sahr, "Hexagonal discrete global grid systems for geospatial computing," *Archives of Photogrammetry, Cartography and Remote Sensing*, vol. 22, pp. 363–376, 2011.
[14] B. Kamgar-Parsi, B. Kamgar-Parsi, and W. A. Sander, "Quantization error in spatial sampling: Comparison between square and hexagonal pixels," in *Proc. of the Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Jun 1989, pp. 604–611.
[15] J. Snyder and D. Mitchell, "Sampling-efficient mapping of spherical images," Microsoft Research, Tech. Rep., 2001.
[16] P. Cozzi and K. Ring, *3D Engine Design for Virtual Globes*, 1st ed. Boca Raton, FL, USA: CRC Press, 2011.
[17] H. Alborzi and H. Samet, "Augmenting SAND with a spherical data model," in *Proc. of the International Conference on Discrete Global Grids*, 2000, pp. 1–19.
[18] R. G. Gibb, "The rHEALPix discrete global grid system," *IOP Conference Series: Earth and Environmental Science*, vol. 34, no. 1, pp. 1–8, 2016.
[19] A. Mahdavi-Amiri, F. Bhojani, and F. F. Samavati, "One-to-two digital Earth," in *Proc. of the International Symposium on Visual Computing*, 2013, pp. 681–692.
[20] D. White, "Global grids from recursive diamond subdivisions of the surface of an octahedron or icosahedron," *Environ. Monit. Assess.*, vol. 64, no. 1, pp. 93–103, 2000.
[21] M. F. Goodchild and Y. Shiren, "A hierarchical spatial data structure for global geographic information systems," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 1, pp. 31–44, 1992.
[22] J. Ben, X. Tong, and R. Chen, "A spatial indexing method for the hexagon discrete global grid system," in *Proc. of the 18th International Conference on Geoinformatics*, 2010, pp. 1–5.
[23] F. E. Wickman, E. Elvers, and K. Edvarson, "A system of domains for global sampling problems," *Geogr. Ann. Ser. B, Physical Geography*, vol. 56, no. 3/4, pp. 201–212, 1974.
[24] P. Peterson, "Close-packed, uniformly adjacent, multiresolutional, overlapping spatial data ordering," US Patent 8400451 B2, 2003.
[25] A. Vince and X. Zheng, "Arithmetic and Fourier transform for the PYXIS multi-resolution digital Earth model," *International Journal of Digital Earth*, vol. 2, no. 1, pp. 59–79, 2009.
[26] G. H. Dutton, *A Hierarchical Coordinate System for Geoprocessing and Cartography*, ser. Lecture Notes in Earth Sciences. Springer-Verlag Berlin Heidelberg, 1999.
[27] A. S. Szalay, J. Gray, G. Fekete, P. Z. Kunszt, P. Kukol, and A. Thakar, "Indexing the sphere with the hierarchical triangular mesh," Microsoft Research, Tech. Rep. MSR-TR-2005-123, 2005.
[28] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, "HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere," *Astrophys. J.*, vol. 622, no. 2, pp. 759–771, 2005.
[29] A. Vince, "Indexing the aperture 3 hexagonal discrete global grid," *J. Vis. Commun. Image Represent.*, vol. 17, no. 6, pp. 1227–1236, 2006.
[30] R. Sadourny, A. Arakawa, and Y. Mintz, "Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere," *Monthly Weather Review*, vol. 96, no. 6, pp. 351–356, 1968.
[31] J. Thuburn, "A PV-based shallow-water model on a hexagonal-icosahedral grid," *Monthly Weather Review*, vol. 125, no. 9, pp. 2328–2347, 1997.
[32] X. Tong, J. Ben, and Y. Wang, "A new effective hexagonal discrete global grid system: Hexagonal quad balanced structure," in *Proc. of the 18th International Conference on Geoinformatics*, 2010, pp. 1–6.
[33] J. P. Snyder, *Map Projections — A Working Manual*. U.S. Government Printing Office, 1987.
[34] ——, *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, 1997.
[35] E. Grafarend and F. Krumm, *Map projections: Cartographic information systems*. Springer-Verlag Berlin Heidelberg, 2006.
[36] H. Hauser, "Generalizing focus+context visualization," in *Scientific Visualization: The Visual Extraction of Knowledge from Data*, ser. Mathematics and Visualization, G.-P. Bonneau, T. Ertl, and G. Nielson, Eds. Springer Berlin Heidelberg, 2006, pp. 305–327.
[37] T. Taerum, M. C. Sousa, F. F. Samavati, S. Chan, and J. R. Mitchell, "Real-time super resolution contextual close-up of clinical volumetric data," in *Proc. of the Joint Eurographics – IEEE VGTC Symposium on Visualization*, ser. EuroVis. Eurographics Association, May 8-10 2006, pp. 347–354.

[38] W.-H. Hsu, K.-L. Ma, and C. Correa, "A rendering framework for multiscale views of 3D models," in *Proc. of the SIGGRAPH Asia Conference*. New York, USA: ACM, 2011, pp. 131:1–131:10.
[39] Y.-S. Wang, C. Wang, T.-Y. Lee, and K.-L. Ma, "Feature-preserving volume data reduction and focus+context visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 2, pp. 171–181, Feb 2011.
[40] E. R. S. Hodges, *The Guild handbook of scientific illustration*. Hoboken, NJ, USA: John Wiley and Sons, 2003.
[41] J. F. Packer, M. Hasan, and F. F. Samavati, "Illustrative multilevel focus+context visualization along snaking paths," *Vis. Comput.*, pp. 1–16, 2016.
[42] J. F. Packer, "Focus+context via snaking paths," Master's thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, Jun 2013. [Online]. Available: http://hdl.handle.net/11023/755
[43] T. Ropinski, I. Viola, M. Biermann, H. Hauser, and K. Hinrichs, "Multimodal visualization with interactive closeups," in *Proc. of the Theory and Practice of Computer Graphics Conference*. Eurographics Association, 2009, pp. 17–24.
[44] M. Cossalter, O. J. Mengshoel, and T. Selker, "Multi-focus and multi-level techniques for visualization and analysis of networks with thematic data," in *Proc. of the SPIE Conference on Visualization and Data Analysis*, vol. 8654, Feb 2013, pp. 1–15, 865403.
[45] Y. Tu and H.-W. Shen, "Balloon focus: A seamless multi-focus+context method for treemaps," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 6, pp. 1157–1164, 2008.
[46] E. Mendez, D. Kalkofen, and D. Schmalstieg, "Interactive context-driven visualization tools for augmented reality," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, Oct 2006, pp. 209–218.
[47] D. Kalkofen, E. Mendez, and D. Schmalstieg, "Interactive focus and context visualization for augmented reality," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, Nov 2007, pp. 191–201.
[48] M. Hasan, F. F. Samavati, and C. Jacob, "Multilevel focus+context visualization using balanced multiresolution," in *Proc. of the International Conference on Cyberworlds*. IEEE Computer Society, Oct 6-8 2014, pp. 145–152.
[49] ——, "Interactive multilevel focus+context visualization framework," *Vis. Comput.*, vol. 32, no. 3, pp. 323–334, 2016.
[50] M. Hasan, F. F. Samavati, and M. C. Sousa, "Balanced multiresolution for symmetric/antisymmetric filters," *Graph. Models*, vol. 78, pp. 36–59, 2015.

Mark Sherlock received his BSc degree in Computer Science from the University of Calgary in 2012, where he is currently working toward his MSc degree in Computer Science. He works alongside *the PYXIS innovation inc.* in Calgary, Canada, assisting in the development of their browser-based Digital Earth visualization. PYXIS products comprise a new generation of decision-making tools that support efficient geospatial analytics and visualization. Mr. Sherlock's research interests include interactive scientific visualization, real-time streaming, computer vision, and network security.

Mahmudul Hasan is a PhD candidate in the Department of Computer Science at the University of Calgary, where he received his MSc degree in Computer Science in 2009. He graduated with a BSc degree in Computer Science from the Independent University, Bangladesh (IUB), as a *Summa Cum Laude*, ranking 1st among the graduating class of 170 students from his school in 2005. Mr. Hasan was the *Posters Chair* of the 39th Graphics Interface (GI) conference in 2013 and the *Organizing Co-chair* of the 3rd and 4th Workshops on Digital Earth held in Banff, Canada in 2015 and 2016, respectively. His research interests include multiresolution and visualization in computer graphics.

Faramarz Samavati is a Professor of Computer Science at the University of Calgary. His research interests include computer graphics, geometric modeling, visualization, and 3D imaging. He has published more than 100 papers, one book, and holds three patents. He is an Associate Editor of Elsevier's Computers & Graphics journal. In the years of 2011 to 2015, Dr. Samavati has received five Best Paper Awards, a Digital Alberta Award, a Great Supervisor Award, and an award from the University of Calgary that recognizes his achievements in innovation, entrepreneurship, and technology transfer.